

Serial No. 09/536,078

- 8 -

Art Unit: 2126

REMARKS

This paper is responsive to the Office Action dated February 24, 2004. All rejections and objections of the Examiner are respectfully traversed. Reconsideration is respectfully requested.

At paragraphs 2-3 of the Office Action, the Examiner rejected claims 1-33 as being obvious under 35 U.S.C. 103, citing the combination of United States patent number 5,983,274 of Hyder et al. ("Hyder et al.") and United States patent number of Dobbins et al. 5,509,123 ("Dobbins et al."). Applicants respectfully traverse this rejection.

Hyder et al. disclose a system for creation and use of control information associated with packetized network data by protocol drivers and device drivers. The Hyder et al. system is designed to allow a software component that processes network data to communicate control information to, and cooperate with, another software component by associating control information with a packet of network data. The Hyder et al. system associates control information with network data upon which control information will operate by appending one or more control data structures to a packet descriptor that is common to all software components processing the network data. The control data structure in Hyder et al. is "tagged" with a class ID value that allows other software components to recognize and utilize the control information. Hyder et al. intend their system to enable any software component to cooperate with and communicate to another software component that processes the network data regardless of any intervening software components.

Hyder et al. use the term "direct call linkage" to refer to a function call interface in their system. As discussed by Hyder et al., address resolution may be done at compile time through traditional linker programs, or may be done dynamically by system components when using such

Serial No. 09/536,078

- 9 -

Art Unit: 2126

entities as dynamic link libraries or export libraries. Hyder et al. teach that network data may contiguously reside in a memory buffer or may be accessed by indirect means or structures such as Memory Descriptor Lists (MDL's) that map segments of virtual memory and their physical page size, and that memory may be requested by a transport protocol driver or a network device driver through an interface call to an integrating component which in turn manages allocating memory. In this way, Hyder et al. teach that network data is made accessible from a packet descriptor.

The packet data structure of Hyder et al. also includes a pointer to a series of control data structures. Each control data structure in the series of control data structures of Hyder et al. is created by an originating software component, such as the transport protocol driver, the network card device driver, or other component that in some way processes the packet, and is filled with control information that will eventually be used by another software component other than the component creating the data structure as the packet descriptor, including all data referenced therefrom, is passed or sent to other software components by means of the integrating software component. Thus the packet descriptor of Hyder et al. functions as an integrating data structure to tie together the network data and the series of control data structures.

Dobbins et al. disclose an object-oriented architecture for network layer routing, which distributes function and system behavior into autonomous router software objects. Dobbins et al. describe distributing functionalities into each object, so that services and data normally external to an object are imbedded or accessible within the object itself. Dobbins et al. further teach that a separate forwarding engine may be provided at each network interface of a networking device. Objects such as these separate forwarding engines are described by Dobbins et al. as having (1) common, protocol-independent functions that are shared by all objects of that class; (2) their own

Serial No. 09/536,078

- 10 -

Art Unit: 2126

configuration information; (3) accessibility through a router resource object for instantiation and control; (4) automatic persistence in NVRAM; (5) remote management capabilities; and (6) text names for navigation of a resource tree as a file system. Dobbins et al. provide that such capabilities are in every object regardless of the specific protocol or application, in order to ensure a common architecture among many different systems/router components, a common method of control internally, a consistent order of instantiation and a common functional behavior.

The router objects of Dobbins et al. have three types of imbedded functionality automatically built in, including 1) common protocol-independent functions of the object, which may be a routing function or a system function, 2) functions provided by a base resource object class which defines methods and data for configuration and control, and 3) functions provided by the managed object class which define the methods and data for network management.

FIG. 3A of Dobbins et al. shows a three-dimensional view of a system architecture including logical relationships between objects, and wherein each functional subsystem is shown as a separate "plane." Thus, this system architecture disclosed by Dobbins et al. FIG. 3A includes four horizontally disposed planes: 1) routing applications, 2) host communications, 3) forwarding, and 4) network interfaces. Multiple object instantiations can exist in the Dobbins et al. FIG. 3A system, such as forwarding and routing protocol objects instantiated for each individual protocol. Connections between the different planes of the Dobbins et al. system are described as logical connections through resource objects and a naming tree.

In FIG. 4 of Dobbins et al., distributed autonomous forwarding engines are used instead of a single centralized forwarding engine. Each of the forwarding engines of the Dobbins et al. system knows how to receive and transmit packets on its own interface, as well as its own

Serial No. 09/536,078

- 11 -

Art Unit: 2126

configuration information, and how to receive and transmit packets on the associated interface, based on a common forwarding table. The specific goal of the each forwarding engine in the Dobbins et al. system is to provide the reception, processing, and forwarding of network layer packets. Accordingly, all forwarding engines in the Dobbins et al. system perform the same basic tasks regardless of protocol or media. In this regard, Dobbins et al. describe a service function for in-bound processing of a network layer packet including reception of the packet from the data link layer, validation and error processing, filter processing, and route lookup processing to determine a next hop. A forward function in Dobbins et al. covers the actual out-bound processing of a network layer packet, consisting of filter processing, converting network layer address to physical address, and passing the packet to the data link layer of the outbound interface for transmission.

FIG. 8 of Dobbins et al. shows a MIB (Management Information Base) structured as a tree, with each node representing an identifier in the Object Identifier (OID). Managed Objects are placed at leaf nodes by the Dobbins et al. system, and can be accessed by name or in sequence in an authenticated manner. Dobbins et al. include no reference to Application Programming Interfaces (APIs) of any kind.

Nowhere in the combination of Hyder et al. and Dobbins et al., taken either individually or in combination, is there disclosed or suggested any:

*... application programming interface to a forwarding plane for processing data packets in a network device that forwards packets across a network, the application programming interface comprising:*

*an input module that receives routing platform independent function calls, wherein the function calls include routing platform independent input control data; at least one control module that receives the input control data via the function calls, the at least one control module producing routing platform specific output control*

Serial No. 09/536,078

- 12 -

Art Unit: 2126

*data based upon the input control data, the output control data being capable of controlling execution of the forwarding plane; and*  
an output module that forwards the output control data from the at least one control module.

as in the present independent claims 1, 9, 18, 21 and 29. Applicants respectfully urge that the interfaces described in the combined references, and cited by the Examiner, are far different from the claimed invention set forth in the present independent claims. Specifically, an Application Programming Interface (API) is described by Hyder et al. as a set of subroutines provided by an integrating driver software component, so that relevant services *in device drivers* may be uniformly accessed *by protocol drivers*. Thus the APIs described in Hyder et al. are used in the *integrating driver that interfaces between a protocol driver and a device driver*. See Claims 1, 8 and 20 of Hyder et al. Hyder et al. include no other reference to APIs. Dobbins et al. come no closer to disclosing or suggesting the above highlighted features of the present independent claims. Dobbins et al. teach protocol-specific Forwarding and Service (FAS) engines derived from a common base class, each protocol forwarding engine of Dobbins et al. having a generic interface for packet servicing and forwarding, regardless of the specific protocol type. In contrast to the presently claimed API features, Dobbins et al. teach a service in which each protocol FAS *registers a pointer to its base class with a packet dispatcher at the data link level for each interface it is instantiated on*. This allows a *packet dispatcher* in the Dobbins et al. system to invoke an overloaded virtual service method without having to know protocol FAS specifics. The combined teachings of Hyder et al. and Dobbins et al. accordingly provide no hint or suggestion of even the desirability of having a system or method including an *application programming interface to a forwarding plane for processing data packets in a network device that forwards packets across a network*, in which an input module receives routing platform

Serial No. 09/536,078

- 13 -

Art Unit: 2126

independent function calls including routing platform independent input control data, and a control module that *receives the input control data via the function calls and produces routing platform specific output control data based upon the input control data*, the output control data being capable of *controlling execution of the forwarding plane*, as in the present independent claims 1, 9, 18, 21 and 29.

For the reasons stated above, Applicants respectfully urge that the combination of Hyder et al. and Dobbins et al. does not disclose or suggest all the features of the present independent claims 1, 9, 18, 21 and 29. Accordingly, Applicants respectfully urge that the combination of Hyder et al. and Dobbins et al. does not support a *prima facie* case of obviousness under 35 U.S.C. 103 with respect to claims 1, 9, 18, 21 and 29. As to the remaining claims, they each depend from claims 1, 9, 18, 21 and 29, and are respectfully believed to be patentable over the combination of Hyder et al. and Dobbins et al. for at least the same reasons. Reconsideration of all pending claims is respectfully requested.

As all claims are believed to be allowable, the application is believed to be in condition for allowance. Withdrawal of the claim rejections and favorable action are respectfully requested.

Applicants have made a diligent effort to place the claims in condition for allowance. However, should there remain unresolved issues that require adverse action, it is respectfully requested that the Examiner telephone the undersigned Attorney at 978-264-6664 so that such issues may be resolved as expeditiously as possible.

Serial No. 09/536,078

- 14 -

Art Unit: 2126

For these reasons, and in view of the above amendments, this application is now considered to be in condition for allowance and such action is earnestly solicited.

Respectfully Submitted,

MAY 24 2004  
Date

David A. Dagg  
David A. Dagg, Reg. No. 37,809  
Attorney/Agent for Applicant(s)  
Steubing McGuinness & Manaras LLP  
125 Nagog Park Drive  
Acton, MA 01720  
(978) 264-6664

Docket No. 2204/A34 120-209